07

Conversiones

Transcripción

[00:00] Ahora vamos a ver cómo convertir estos tipos de datos entre ellos, cómo hacerlos compatibles de cierta forma. Nuevamente vamos a borrar esto y por ejemplo, ¿qué pasaría si yo tengo una variable double variable1 igual 230.89, y yo quisiera transformarla para un entero?

[00:44] ¿Cómo podría hacer? Es muy simple. Tenemos en Java lo que se llama cast. ¿Qué es cast? Si yo asigno variable1 aquí, él no va a compilar, pero yo puedo forzar esto. ¿Cómo? Abriendo un par de paréntesis aquí especificando el tipo de dato al que yo quiero convertir variable1. Aquí yo estoy forzando que variable1 sea un entero, estoy obligándolo a ser un entero.

[01:22] Y de la misma forma que vimos en el ejemplo de la división, si yo aquí hago un system.out.println e imprimo variable entero, salvamos aquí, damos play, y vamos a ver que él imprime la parte entera, 230, mas él olvida la parte decimal. Él no va a hacer aquí un redondeo ni nada de esto, él simplemente va a va a ignorar completamente la parte decimal y nos va a dar solamente la parte entera.

[02:02] Ahora, si nosotros realmente deseamos ese resultado, hacer un cast nos vendría muy bien. Este ejemplo ese un cast, estamos casteando la variable para cumplir con la condición que nosotros queremos que en este caso es ser un tipo entero.

[02:24] Ahora, ¿se dieron cuenta que aquí hice estas dos líneas, escribí cast y él no generó ningún error en el código? Esto se llama un comentario. Con estos

dos slash al frente, nosotros hacemos comentarios en Java, podemos tener comentarios para describir el código, por ejemplo: este método imprime en consola.

[02:55] Y podemos comentar código incluso. Si no queremos que este código sea ejecutado, podemos simplemente ejecutarlo y él va a ser ignorado en su totalidad por el compilador y por ende no va a tener ejecución alguna. Si nosotros guardamos y corremos aquí el programa vamos a ver que no tiene nada porque hemos comentado el statement de System.out.println.

[03:25] Entonces, hablando ya acerca de los tipos de variables que ya hemos visto, double e int, entrando un poco más a fondo en la explicación por qué double soporta digamos un tipo de número decimal y por qué int no, esto es porque int soporta una cantidad de 32 bits. ¿Esto qué significa?

[03:53] Que int soporta como máximo un valor de 2 elevado a la 31 para positivo y 2 elevado a la 31 en negativo, menos 1, porque tenemos el 0 al medio. Entonces nosotros no podemos asignar un valor mayor a 2 a la 31 a una variable del tipo int por ejemplo, int. Prueba. Si nosotros queremos dar un valor muy grande a un int, él nos va a dar un error de compilación, mismo siendo un valor entero.

[04:30] ¿Por qué? Porque él, este número, que ni siquiera sé cómo se pronuncia, cuál es el nombre exacto, está fuera del rango de int, fuera del rango de bits de int y no soporta este tamaño de número. Entonces, ¿qué usamos en este caso? Usamos otro tipo de dato que se llama long. Long soporta 2 a la 64 bits.

[04:59] Perdón, soporta 64 bits y 2 a la 63 positivo, 2 a la 63 negativo menos 1, porque tiene el 0 al medio, obviamente. Entonces, nosotros para asignar tremendo número a un long, especificamos para el compilador que él va a ser un número del tipo long, y aún así este número escapó del rango total que abarca long.

[05:26] Vemos que aquí, si acortamos un poco más el número, que aún así no sé cómo sería pronunciarlo, ya entra dentro de un tipo long, pero aún sí este número sería muy grande para un tipo int. ¿Por qué la L al final? Porque necesitamos especificar que el tipo de dato de este número, en este caso estamos seguros que es un número grande, es un long, y lo especificamos ya sea con L mayúscula o L minúscula.

[05:56] Si no especificamos eso, nos da un error de compilación. Entonces, en mi caso a mí me gusta especificarlo con L mayúscula porque es mucho más notorio a la vista. Muchas veces si es L minúscula, puede ser confundido con un 1 y eso no es lo que queremos. La idea es escribir código que sea fácil de leer para las personas, no para las máquinas.

[06:18] Ahora, así como tenemos un tipo de datos que soporta números gigantes, también tenemos tipos de datos que soportan números más pequeños, este es el caso de short. Short es un tipo de dato que es para números pequeños que va en un rango de 2 a la 16, entonces él va a soportar un número así de pequeño.

[06:55] Y tenemos incluso un tipo de dato que soporta números más pequeños, como byte, que él soporta solamente 8 bits de tamaño, entonces byte es número aún más pequeño. Dense cuenta, noten esta notación que yo estoy usando, es llamada camel case porque es tipo como un camello, como si las mayúsculas fueran las jorobas de un camello. Es la notación más usada aquí en Java.

[07:31] Por ejemplo, para especificar nombres de clase hemos visto que se comienza con letra mayúscula. Hay otros lenguajes de programación donde se usa letra minúscula y no hay ningún problema. Hay lenguajes de programación donde se acostumbra declarar las variables con mayúscula y es totalmente válido. Es posible en Java, pero no es la buena práctica, no es lo que se hace normalmente, hablando solamente dentro de Java.

[07:58] En el caso de aquí siempre acostumbramos declarar nuestras variables de forma que quede entendible para cualquier persona qué representa aquella variable. Por ejemplo, yo no podría hacer algo como numMinPeq, por ejemplo. Número mínimo pequeño. Quizás yo sé qué significa esto, pero alguien que viene a leer mi código, él no va a entender a qué estoy haciendo referencia con este nombre.

[08:35] Entonces por eso somos bien explícitos al declarar nuestras variables. Número aún más pequeño. Y él va a soportar sus 8 bits de tamaño. Entonces, no se preocupen mucho por estos dos tips de variable, es muy extraño usar ellas. Vamos a usar bytes más que nada cuando estemos haciendo procesamiento de archivos, archivos de texto o lo que sea. Y short es muy raro que lo vean.

[09:13] Otro tipo de dato que tenemos numérico es float. Float es un tipo de dato parecido a double que también soporta decimales pero también es más pequeño, entonces este es un número decimal pequeño y este puede ser un 2.5, pero para especificar qué es un float, de la misma forma que lo hacemos con long, agregamos aquí una F y listo. Con esto ya tenemos compilación correcta del valor asignado a cada tipo de variable.

[09:58] Nuevamente, estos tres últimos tipos no los vamos a ver a lo largo del curso. Long lo vamos a ver muy poco, solo en algunos casos muy específicos. Vamos a centrarnos más que nada en double e int que son los más usados actualmente en casi todos los programas. Long es usado a veces, no es muy común, y esto es usado menos común.

[10:28] No se usa generalmente para lo que es números sino ya para lo que es valores de otro tipo que ya vamos a ir viendo en algunos cursos o tópicos más avanzados en Java. Ya con esto tenemos los tipos de variables un poco más entendidos. Este es el primer paso ya para comenzar a escribir código un poco más complejo cada vez, si se dan cuenta.

[10:54] Ya hemos escrito código que vamos a encontrar digamos en cualquier proyecto de programación y ya tenemos una idea de cómo podemos seguir usando estas variables. Por ejemplo, podemos sumarlas, podemos sumar variable1 + variable1 entero y esto asignarlo a un int resultado. ¿Por qué él me está dando un error de compilación?

[11:30] Porque recordemos que variable1 es un decimal y no va a entrar en este int, a diferencia de variable1 entero que ya está casteado. Entonces aquí podemos simplemente cambiar el tipo de valor y resuelve, porque ya nuevamente, haciendo un pequeño énfasis, el entero entra aquí dentro de un tipo double, o nuevamente hacemos esto y casteamos el resultado.

[12:05] Pero ya sabemos que obviamente nos arriesgamos a perder precisión. Pero si aún así estamos seguros de lo que queremos hacer, imprimimos. Nuevamente corregimos acá, salvamos, ejecutamos y tenemos el resultado. Con eso ya tenemos un poco más claro cómo funcionan los tipos de datos e Java y ya vamos a ver más adelante cuál es la gran utilidad que tiene esto.

[12:47] Quizás ustedes pueden decir: "pero en JavaScript o en otro lenguaje de programación yo no necesito declarar el lenguaje, automáticamente entiende qué tipo de dato yo estoy haciendo referencia". No se preocupen, a lo largo que avancemos con el curso vamos a ver todas las ventajas que tiene Java.